Topic Selection in Industry Experiments

Ayse Tosun Misirli
[1]Department of Information Processing Science
University of Oulu
Oulu, FI-90014, Finland
ayse.tosunmisirli@oulu.fi

Hakan Erdogmus
Carnegie Mellon University
Silicon Valley, Building 23
Mountain View, CA, USA
hakan.erdogmus@sv.cmu.edu

Natalia Juristo[1,2], Oscar Dieste
[2]Facultad de Informática
Universidad Politécnica de Madrid
Madrid, Spain
{natalia, odieste}@fi.upm.es

ABSTRACT

This paper shares our experience with initial negotiation and topic elicitation process for conducting industry experiments in six software development organizations in Finland. The process involved interaction with company representatives in the form of both multiple group discussions and separate face-to-face meetings. Fitness criteria developed by researchers were applied to the list of generated topics to decide on a common topic. The challenges we faced include diversity of proposed topics, communication gaps, misunderstanding about research methods, initial disconnect between research and industry needs, and lack of prior work relationship. Lessons learned include having enough time to establish trust with partners, importance of leveraging the benefits of training and skill development that are inherent in the experimental approach, uniquely positioning the experimental approach within the landscape of other validation approaches more familiar to industrial partners, and introducing the fitness criteria early in the process.

1. INTRODUCTION

Experiments conducted with industrial professionals are often considered to produce more generalizable results than their counterparts run in academia [6, 7]. They contribute to building scientifically confirmed theories in software engineering by testing the cause-effect relationships that were previously studied through laboratory experiments (in academia). However, the majority of the experiments conducted in the software engineering context have been in academia.

A systematic literature review reveals that only four industry experiments were conducted until 2002, and another 11 were conducted between 2003 and mid-2012 [8].

1.1 Challenges with Industrial Experiments

In experiments conducted in controlled (laboratory) settings where the technical infrastructure for monitoring subjects and collecting data are provided by researchers, using students as subjects is the norm. In these environments, students can be motivated through a reward or grading system. When needed, an initial training on the treatments can be given in the scope of the curriculum. Alternatively only experienced students can be selected. Such options make it relatively easier to conduct experiments in an academic setting.

Conducting experiments in industry introduces additional challenges for researchers from planning to reporting. Industry experiments are subject to many confounding factors that may threaten their validity even though they increase the generalizability of the results. For example, when the subjects are software professionals in a company, researchers may not be able to control subject selection and their profiles, resulting in pure convenience sampling. Furthermore, researchers cannot easily enforce adherence to a treatment in a setting that is completely unfamiliar to the subjects. Motivational factors also come into play in a significant way. For both professionals who serve as subjects and their managers, tangible benefits associated with participating in an experiment are fewer, and consequently, buy-in is much more difficult than with students.

Experimentation in the industry is generally more costly than in academia since the tasks, subjects and the environments of the experiments should be as realistic as practically possible [6]. While these requirements translate into challenging experimental designs and more effort on the part of the researchers, there might be a significant associated cost to the companies in terms of diverted resources. It is thus difficult to convince software professionals and their employers to spend their valuable and limited resources to contribute to an experiment.

1.2 Mutual Understanding of Needs

Successful industry and academia collaborations rely on joint and early agreement of mutual goals and roles. Researcher should understand the industry's reasons to participate in the collaboration [5]. They should be aware of the challenges of their industrial partners, and choose problems and topics that are novel, feasible, industrially relevant, and potentially impactful [4]. If the problem or topic is sufficiently motivating, buy-in can be secured and the chances for a successful collaboration increases.

Existing guidelines for conducting experiments in software engineering usually miss the importance of initial negotiation and topic selection as a main driver. Wohlin et al. [10] state that "the

starting point for an experiment is insight, and the idea that an experiment would be a possible way of evaluating whatever we are interested in". Similarly, Kitchenham et al. [11] advocate the experimental context as the first step of an experiment, during which the objectives of the research are properly defined and the description of the research provides enough detail for researchers and practitioners. The process of finding a relevant topic that is linked to a real challenge and tangible outcomes in the industry, and that is suitable for experimentation is not given much upfront attention.

The identification of a suitable problem or topic as a main driver is however implied by Jedlitschka and Pfahl [12], who partition the definition of experimental context into problem statement, objective, and context. They argue that the problem and its relevance to the context need to be justified before defining research objectives. This argument aligns with our perspective.

The centrality of topic selection was the premise of our industry-academia collaboration in Finland. We recognized early on in the project that topic selection could be paramount in overcoming motivational barriers for our industrial partners. Our first priority was therefore to identify software engineering topics that are both interesting and attractive for practitioners and appropriate for studying with an experimental approach.

In the rest of the paper we share our process and experience with initial negotiation and topic elicitation process in the context of our collaboration with six Finnish software organizations. We report on the steps that we followed in this process, which involved face-to-face meetings with software professionals, forming an initial list of topics that are of interest to our partners, assessment of each topic for its fitness for study using an experimental design, and final selection of an initial and agreed-upon topic to pursue. We also discuss the challenges faced during the topic elicitation process, and follow up the challenges with a set of lessons learned. These lessons will help us improve our future industrial collaborations and justify the value of experiments in their organizational contexts.

2. TOPIC ELICITATION PROCESS

2.1 Context

Our experience with selecting software engineering topics suitable for experimentation within companies started in early 2013 in the context of the Experimental Software Engineering Industry Laboratory (ESEIL) project. The ESEIL project aims to help Finnish companies achieve efficiency and transparency in their software development processes through experimentation. The operating hypothesis of ESEIL is that if the companies understand the impact of using new practices and techniques that are of interest to them in their own context, they will be able to make better decisions regarding their adoption. With that goal in mind, the researchers approached the ESEIL contributors to identify topics that would provide value to companies and in turn, are worthy of study. We were specifically looking for a set of common topics that would be of interest to all the companies so that we could maximize the utilization of study designs and experimental packages, and pool the data from multiple organizations for meaningful analysis. We needed to utilize limited research resources to the best of our ability. The immediate objective was to identify one viable topic for the first phase of ESEIL experiments.

The first task was to inform the companies on the project motivation, goals, elements of the scientific approach used, and potential benefits. This allowed us to generate sufficient interest to continue. We also needed to determine how to gain access to a representative population of the software development organizations so that we can have meaningful input on which topics to pursue as the subject of the first batch of investigations.

2.2 Kick-off meeting

The process started in January 2013 with a kickoff meeting with representatives from eight companies. The kickoff meeting was run as a workshop. The lead researcher explained the purpose of experimentation in software engineering, gave an overview of the methods used, provided comparisons to other disciplines, listed common topics pursued by empirical software engineering researchers, and suggested a plan to move forward.

The original idea was to solicit topics by deploying a short survey within each participating software development organization. However, this option was deemed too obtrusive at the kickoff meeting and proved infeasible. We attributed the companies' reluctance to dedicate widespread employee resources at these initial stages to their lack of prior working relationships with us. Invariably, the company representatives preferred to have face-to-face meetings with a few selected employees to explore possible topics of common interest. Separate face-to-face meetings were also seen as an opportunity to better understand the advocated research approach and to have further conversation to seed the topic selection process using a more guided format. Out of the eight companies that attended the kickoff meeting, six decided to follow up the proposed collaboration with separate face-to-face meetings involving multiple company representatives. The other two dropped out due to internal problems. We asked our contacts from the six companies that agreed to participate to invite several employees with different software development roles and experience to ensure broad representation.

2.3 Face-to-Face Brainstorming Meetings

The profiles of the six companies and their representatives that participated in the separate topic elicitation meetings are summarized in Table 1. We expected to have a larger number of company representatives at each meeting, but our contacts deliberately wanted to keep the meetings small to reduce the impact on their business.

The face-to-meetings were conducted as unstructured group interviews. Multiple researchers participated in each meeting, with one researcher leading and asking questions and one researcher taking notes. The companies were first asked to comment on the types of software they develop and the technologies, practices, and processes they use to develop the software. These questions set the context, and the participants were encouraged to talk about specific issues and interests from their own roles' perspectives. The following question seeded the discussion: "What kind of questions would you like to answer to help you improve the way your teams develop software, the way the development teams interact with their respective clients, and the quality of the resulting software?"

Potential topics were elicited in this group setting, and the topics that were both mentioned in passing and explicitly proposed by the participants were recorded by the scribe. The recorded topics

from all meetings were then aggregated in a single list. Closely related topics were merged using ad-hoc iterative clustering.

Each face-to-face meeting took about an hour, and in some cases exceeded an hour. The actual length depended on the number of participants and how engaged and interested they were. We essentially left it to the company representatives to figure out their own organizations' needs when proposing candidate topics. The researchers avoided further guidance to minimize bias. All of the companies visited had a keen interest in agile and lean software development; five were already applying agile and lean practices regularly. Thus many of the topics proposed had some link to these paradigms.

Table 1. Profiles of companies participated in the process

	Size[1]	Domain	Company representatives		
			#	Roles	Seniority
A	Large	Security	2	Innovation Manager	Senior
B	Small	CRM	5	Developer, Architect, Product Owner	Mid-level
C	Med.	Games	3	Manager, Technical Lead, UI Expert	Senior
D	Large	Embedded	3	Manager, Developer, Q/A Lead	Senior
E	Large	Telecom	3	Manager, Q/A Lead, Testing Lead	Senior
F	Small	Retail oil	1	Manager	Senior

[1]Large: > 500 employees; Medium (Med.): 100-500 employees: Small: < 100 employees

2.4 Topic Fitness Criteria

In selecting a suitable topic, a balance must be achieved between researcher and industry needs. This tradeoff is difficult since the two perspectives may clash. Not all software engineering topics lend themselves easily to study using an experimental approach. Others may require intimate domain knowledge or substantial previous experience. In addition, topics differ in terms of their relevance to the research community, characterized by general worthiness of scientific inquiry (beyond the interests of a single company and transient popularity at a particular point in time) and publishability of results. The approach we took relied first on identifying company interests to ensure industrial relevance. Research relevance and feasibility were considered in the second step by filtering these interests based on the four fitness criteria:

1. Concreteness: If the company's need and its underlying research question are not sufficiently well-defined, more time will be needed by the research team to dissect the problem, reduce the problem to a concrete form, and come up with a suitable experimental design. The risk of failing to satisfy the company's need increases if the topic is too abstract.

2. Suitability for Experimentation: Certain software engineering topics are subject to multiple intangible factors with long-term and organization-wide implications, and as such are better fits for longitudinal field studies with a qualitative orientation. These topics may be difficult to study using experimentation in contrast to topics such as automatic test case generation. For example, it is particularly challenging to design and conduct experiments for evaluating software engineering techniques where human factors, such as attitudes of users and developers, play a major role and development tasks which need to be monitored more than 2-3 hours a day. Topics for which active involvement and observation of subjects over a long period of time are required also have low feasibility and high cost, and in turn, they might be more suitable for qualitative studies.

3. Relevance to Research Community: The research community values contributions that build on existing knowledge. Topics that have potential to add value to, strengthen, or refute existing results generally reach a wider audience. Consequently, the research designs used and the findings are easier to compare to those of previous studies. Studies that build on others also allow the researchers to leverage lessons learned from previous studies, thereby mitigating validity and execution risks.

4. Prior Experience: Experiments often involve training subjects on a topic or techniques unfamiliar to them, and designing realistic tasks that address problems in the topic's domain. In addition, testing hypotheses, interpreting findings, and drawing meaningful conclusions may require substantial domain knowledge. Thus researchers do not only need experience in experimental design and research methods in general, but also in the topic, and its underlying domain, being studied. Lack of prior domain experience may raise serious threats to construct validity of a study.

These four criteria represent primarily the researchers' constraints. They were applied as a filtering mechanism to candidate topics. Since we used a pull rather than a push approach during meetings with our industry partners, we assume the candidate topics to be already industrially relevant. Our purpose in articulating the criteria was to make our industrial partners aware of the additional tradeoffs that might not have been visible to them.

2.5 Topic Selection Meeting

The separate face-to-meetings with the companies were followed up by a topic selection meeting attended by representatives from all participating companies. The researchers presented the outcome of the face-of-face meetings and explained the four fitness criteria. They next summarized the evaluation of the proposed topics relative to the fitness criteria, in which each topic was rated on a three-point scale with respect to each criterion. Table 2 gives this summary. The ratings had been assigned by researchers prior to this meeting. The highest-rated topics are typeset in bold in Table 2.

The suggested topics exhibited a large variety. Some topics were too general (e.g., open source, testing), whereas some were very platform dependent (e.g., Eclipse versus competitor IDEs). The most frequently mentioned topics were related to quality assurance and testing, while the least frequently mentioned topic was software design (omitted in Table 2). We believe that professionals need some guidance when they are asked about their topics of interest; otherwise they may think too abstractly or in too detail. We didn't specifically investigate why certain topics were suggested and they differed in specificity. It would have been revealing to know the motivational factors behind the suggested topics, e.g., whether the topics correlated with the participants' formal education and roles, or whether the suggestions had more to do with the popularity of topics in the industry. The company representatives discussed the list in Table 2. The factors pertaining to industrial relevance, i.e., topicality, learning opportunities, originality and alignment with development process and interests, were brought up.

Table 2. Topic list and their ratings based on four criteria

	Criteria			
Topic	**Concreteness**	**Suitability**	**Relevance**	**Experience**
Requirements related				
Product definition	Low	Low	Low	Low
Prioritization of tasks	Medium	Medium	Medium	Medium
User Needs (Requirements) Elicitation	Medium	High	High	High
Quality Assurance related				
Testing	Low	High	High	High
Testing team versus beta users	High	Medium	Low	High
Model based testing	Medium	High	High	Low
Test with or without the hardware	High	High	Low	Medium
Manual versus automatic test case generation	High	High	Medium	High
Time need to find bugs	High	High	Medium	High
Influence of programming language on testing	High	High	Medium	Medium
Development related				
Refactoring	Low	Low	High	Low
Test driven development (TDD) versus non-TDD	**High**	**High**	**High**	**High**
Pair Programming	**High**	**High**	**High**	**High**
Process related				
Open source	Low	Low	High	Medium
Product line management	Low	Low	High	Low
Importance of granularity of commits	Medium	Medium	Medium	Low
Kanban for developers at customer site	Medium	Low	Low	Low
Human factors related				
Impact of personality on code reviews	**High**	**High**	**High**	**High**
Overhead of context switching for developers	High	High	Medium	Low
Interaction between developer and HCI experts	Medium	Low	Medium	Medium
Tools				
Eclipse IDE versus competitor IDEs	High	High	Medium	Low
Version control systems: centralized vs. distributed	High	Medium	Low	Low

After the discussion, the company representatives were asked to vote on topics. Each company had three votes, which could be distributed among at most three topics. The representatives who were not present were later given the opportunity to vote offline on the top two topics that garnered the most votes. The top two topics that received the highest number of votes were "Test-Driven Development" and "Requirements Elicitation".

2.6 Selected Topic: Test-Driven Development

The topic that received the most votes was Test-Driven Development, and it was subsequently chosen as the subject of the initial phase of the ESEIL project. Requirements Elicitation topic has still been under negotiation for clarifying the problem better and increase its concreteness score. It was decided that the remaining topics could be re-evaluated and considered as pending experiments in the subsequent phases of the project.

TDD has many noteworthy characteristics that make it both industrially relevant and a good fit for study through experimentation in an industrial context. Since first popularized in the context of Extreme Programming in early 2000s, TDD has continued to be a topical and controversial practice. It advocates interleaving normally distinct activities, such as construction, testing, low-level design, and even documentation, into a micro process that requires skill, mastery, and extra effort by developers. It promises both long-term productivity and quality benefits in return when practiced systematically by the development team [2].

TDD's industrial relevance is evidenced by its consistent high ranking by software development managers [1]. However, due to its reputation as a challenging practice to apply, TDD had been unexplored at the organization-level by our industrial partners, despite the fact that the companies were familiar with and using many other agile practices. One company had had some employees with TDD training, but the practice was not

encouraged or supported organization-wide. These characteristics motivated the companies to use the ESEIL project as an opportunity to try TDD out in their own contexts as well as to augment the skill set of their developers.

In terms of satisfying the fitness criteria, TDD is well-defined enough as a development technique. It lends itself to study using an experimental approach because it's a low-level technique with a natural control alternative, and both its productivity and quality effects and the subjects' level of conformance to the workflow it prescribes are objectively measurable. The quality and productivity effects of TDD have been studied extensively by several researchers [9], which make TDD relevant to the research community. Interestingly, cumulative evidence about TDD's effectiveness is still not conclusive enough [2], with contextual factors appearing to play an important role. This is an additional motivating factor because it promises ample opportunities to extend the existing knowledge. Finally, the research team had substantial collective experience applying TDD, using it, studying it, and teaching it, with an arsenal of experimental designs and results at their disposal to build on. This collective experience gave the research team a unique domain knowledge advantage.

After the topic selection meeting, four companies decided to participate in the TDD experiments. The remaining two decided that the topic didn't fit in with their priorities at the time.

2.7 Challenges

We faced several challenges, particularly in the initial phases of the topic elicitation process, which pertained to diversity of topics, communication gaps, misunderstanding of research methods, disconnect between needs and lack of prior relationship.

Diversity of topics. We expected most topics to be pervasive among the software organizations that we approached. We were surprised that even among the companies following similar type of development methodology and within the same company, many proposed topics were unique. There was no obvious convergence to a common area, and consensus seemed elusive. Some of the topics were so marginal that it led us to think that the employees did not mention more obvious, but hard problems because they didn't have a realistic expectation that an academic approach could effectively tackle them.

Perhaps the diversity of suggested topics should not have been surprising, since the differences among software practitioners in terms of their experience, role, and in turn, their dissimilar interest can lead to this variation. The results of previous surveys at Microsoft on employees' suggestions of interesting software engineering topics also exhibited much variation [4]. These surveys resulted in 12 prioritized categories that spanned a variety of activities and issues with rankings highly dependent on role, geography, and experience level.

Communication gaps. Terminology differences between researchers and practitioners were a pervasive source of misunderstanding. Practitioners sometimes referred to well-known software engineering concepts using company-specific acronyms or terms. Among the companies themselves, similar concepts were sometimes referred to by different names, and clarifications were often required. Vagueness of concepts and translation to English may have compounded these communication problems. In a few instances, we were unsuccessful in recording a proposed topic in concrete and well-known terms.

Misunderstanding of research methods. Although the company representatives initially appeared to grasp the notion of an experiment as a way of evaluating alternative software engineering approaches, and went along with the idea, they later had an issue with the experimental approach. The necessity of using multiple subjects in multiple treatment groups appeared to them as too resource-intensive, logistically and politically complicated, and somewhat wasteful. Although the experimental approach was explained in the beginning, some companies later appeared to be surprised by its implications, and decided not to be involved in the first batch of the experiments.

Disconnect between needs. Based on our past experience, we were well aware of the differences between researcher needs (publications, research relevance, technical feasibility, and validity) and company needs (effort spent for experimentation, tangible benefits to company in terms of skills development, quality and productivity improvement, new knowledge that aids solving a critical problem). Even though we rationalized the research needs during the topic elicitation process using the fitness criteria and explained it, it was still difficult for companies to come to terms with them.

Lack of prior relationship. Skepticism about the concept of an experiment and research methods was apparent from the beginning. Skepticism manifested itself in subtle ways, as silence and lack of enthusiasm in meetings, and reluctance to provide access to employees without reasonable assurance of benefits. Our initial plan to run a survey was to get as much feedback from employees as possible. This idea was rejected in favor of group meetings with a few selective individuals. Lack of a working relationship prior to ESEIL project possibly played a role in the companies' reluctance to provide us access to their employees, since after we had developed that relationship, one of the most reluctant company contacts proposed to solicit input from many more representatives in the future.

3. LESSONS LEARNED

The challenges faced during topic elicitation process led us to rethink our approach. We believe that addressing certain challenges earlier in the process would have helped alleviate some of the mutual frustrations and roadblocks. In retrospect, the initial expectations might have been too high on both sides. Managing those expectations more proactively would have been advisable. Key points to take away from this experience and key changes that we are planning to implement in future rounds of the ESEIL project are summarized below.

Establishing trust takes time. It is not reasonable to expect companies to be enthusiastic about a research-based approach from the very beginning. Energy and effort put into developing and nurturing relationships and creating champions pay off. We were fairly successful in this regard once we overcame the initial hurdles. The resistance faced regarding access to employees was a wake-up call. In the future, a gentler introduction based on one-on-one interactions with the main company contacts (champions) before the kickoff meeting is likely to prove useful. The company champions sometimes emerged later in the process than we would have liked because of delayed interactions with the key personnel.

Understand why a particular topic is of interest to the practitioners. Although we did not ask why an employee proposed a particular topic (we were more interested in the topics themselves than the reasons for suggesting them), we suspected that topic proposals were (a) influenced in terms of the topics'

generality vs. specificity by the employees' role and level inside the organization and (b) motivated in terms of coverage by either a perceived area of weakness within the organization about a topic or a lack of knowledge about a topic coupled with an intuition that more knowledge is needed. In the future, addressing the influencers and motivators explicitly could provide us with better clarity about the topics proposed and more solid grounding in establishing industrial relevance.

Explain researcher needs and the topic fitness criteria earlier in the process. Articulating the fitness criteria and using it explicitly during topic selection was extremely useful in communicating our needs. However, doing this upfront may have set boundaries and filtered out some topics though they would be very interesting and useful for practitioners. We plan to discuss the validity of fitness criteria earlier in brainstorming meetings and argue what could be done for topics that researchers did not have much experience, but were considered as useful by practitioners.

Differentiate the experimental approach from other validation strategies used in industry. The needs, benefits, and caveats of different validation strategies are different, and the companies did not display a clear understanding of the differences, causing them confuse experiments with pilot studies. This confusion could have been avoided if the experimental approach had been positioned within the larger landscape of empirical approaches. In particular, we could have explained the differences between experiments and other empirical approaches, and the benefits of experiments over other controlled validation models used in industry (e.g. feasibility evaluations, pilots) [3].

Leverage agile and lean values to get buy-in. All of the companies were familiar with agile and lean principles, and most had already embraced some relevant key practices. Experimentation is mentioned in several agile and lean approaches in the literature. Thus, we could have leveraged this explicit focus and strengthened the argument for experimentation by appealing to related agile and lean tenets, such as continuous improvement, learning, benchmarking, importance of adaptation to own context, use of scientific method, and measurement.

The training angle gave us the best leverage we had in securing buy-in. Training is often an integral part of the experimental approach while being of significant interest to the companies. It may be central for aligning research and industry interests. Some organizations have set budgets allocated to training and professional development, with employees being required to take advantage of these budgets. This was instrumental in removing management's usual concerns.

4. CONCLUSIONS

In this paper, we describe our experience with initial negotiation and topic elicitation for conducting industry experiments with six companies. We do not aim to identify worthwhile topics for all industry experiments, nor attempts to generalize the results pertaining to an instance of topic elicitation. Based on our experience, we have found that identifying relevant and motivating software engineering topics for conducting industry experiments is difficult. There is considerable variation in the selected topics, both within the same organization and across six organizations. Even in the presence of strong consensus, a topic that is deemed industrially relevant may later turn out to be unviable or a poor fit for an experiment. Researchers should therefore be well served by tackling topic selection and fitness early and systematically in their industrial collaborations. Close collaboration with practitioners, i.e., understanding their needs and attracting them for experimentation, is also as essential in industry experiments as in case studies. Researchers should strive to clarify the novelty and relevance of candidate topics with their industrial collaborators. They should make research needs, methods, and criteria for topic selection transparent to companies.

5. ACKNOWLEDGMENT

This research has been partly funded by Spanish Ministry of Science and Innovation projects TIN2011-23216 and TEKES' Finland Distinguished Professor Program.

6. REFERENCES

[1] K.E. Emam, Finding Success in Small Software Projects, Agile Project Management Executive Report, 4 (11), Cutter Consortium, Arlington, Massachusetts.

[2] B. Turhan, L. Layman, M. Diep, H. Erdogmus, F. Shull, How Effective is Test-Driven Development?, in *Making Software: What Really Works, and Why We Believe It,* A. Oram, G. Wilson (eds.), O'Reilly, 2010.

[3] M.V. Zelkowitz, D.R. Wallace, D.W. Binkley, Experimental validation of new software technology, in *Lecture notes on Empirical Software Engineering*, N. Juristo and A. M. Moreno (Eds.). Software Engineering and Knowledge Engineering, Vol. 12. 2003, 229-263.

[4] A. Begel, T. Zimmermann, Analyze This! 145 Questions for Data scientists in software engineering, Microsoft Research Technical Report, MSR-TR-2013-84, 2013.

[5] C. Wohlin, Empiricial Software Engineering Research with Industry: Top 10 Challenges, in *Proc. of 1St Int. Workshop on Conducting Empirical Studies in Industry,* 2013.

[6] D.I.K. Sjoberg, B. Anda, E. Arisholm, T. Dyba, M. Jørgensen, A. Karahasanovic, E. F. Koren, M. Vokác, Conducting Realistic Experiments in Software Engineering, in *Proc. of 1St Int. Workshop on Conducting Empirical Studies in Industry,* 2013.

[7] L. G. Votta, By the way, has anyone studied any real programmers, yet?, in *Proc. of 9th International Software Process Workshop*, 1994, 93-95.

[8] O. Dieste, N. Juristo, M. D. Martinez, Software industry experiments: A systematic literature review, in *Proc. of 1St Int. Workshop on Conducting Empirical Studies in Industry,* 2013.

[9] Y. Rafique, V.B. Misic, "The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis", *IEEE Trans. on Software Engineering*, 39 (6), 2013, 835-856.

[10] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering, Springer, 2012.

[11] B. Kitchenham, S.L. Pfleeger, L. Pickard, P. Jones, D.C. Hoaglin, K. El Emam, J. Rosenberg, Preliminary Guidelines for Empirical Research in Software Engineering, *IEEE Trans. on Software Engineering*, 28 (8), 2002, 721-734.

[12] A. Jedlischka, D. Pfahl, Reporting Guidelines for Controlled Experiments in Software Engineering, in *Proc. of International Symposium on Empirical Software Engineering,* 2005.